

11 СТВОРЕННЯ ЗАГОЛОВНИХ ФАЙЛІВ

Мета: Здобути практичні навички роботи зі створення та використання заголовного файлу.

11.1 Короткі теоретичні відомості

Якщо певний проект розробляє група програмістів, кожний з яких виконує окреме завдання, має сенс розв'язок кожного завдання розташовувати в окремому файлі, а потім долучати всі такі файли до спільного проекту. Такий підхід дозволяє більш оптимально організувати роботу команди програмістів і більш чітко визначити взаємодію частин програми.

Відповідно до засад технології програмування, великі програми розбиваються на частини за функціональною направленістю. Один файл може містити програмний код підтримки графічного виведення, другий—математичні розрахунки, третій— код введення-виведення даних файла. При написанні великих програм тримати всі ці частини в одному файлі-програмі складно чи взагалі не можливо.

Заголовні файли мають розширення `h` чи `hrr` і містять заголовки (прототипи) функцій, а також оголошення типів, констант та змінних з ключовим словом `extern`. Перевага заголовних файлів полягає у тому, що, коли створено заголовний файл для одного модуля (одного чи декількох `crr`-файлів) програми, можна переносити всі зроблені оголошення, які були зроблені, до іншої програми `C++`. Для цього слід просто залучити заголовний файл за допомогою директиви `#include`. Зазвичай для кожного `crr`-файла створюється власний заголовний файл з таким самим ім'ям і розширенням `h`. І, навпаки, здебільшого, для кожного заголовного файла створюється `crr`-файл з реалізацією функцій, оголошених у `h`-файлі. При залученні до програми заголовного файла його текст (а з ним і текст відповідного `crr`-файла) автоматично вставляється до програми замість рядка з відповідною директивою `#include`.

Заголовні файли можна розділити на стандартні й створювані програмістом.

Стандартні заголовні файли зберігаються у спеціальній теці `INCLUDE`, яка є підтекою теки, в яку було встановлено `VC`. Імена стандартних заголовних файлів пишуться у кутових дужках "`<`" і "`>`", наприклад:

```
#include <math.h> /* Залучення заголовного файла з математичними функціями*/
```

Окрім того, для таких файлів можна не зазначати розширення `h`. Якщо стандартний заголовний файл розміщено в іншій теці, слід зазначити його

місцезнаходження, наприклад, щоб залучити до програми заголовний файл `types.h`, який міститься у теці `INCLUDE\SYS`, слід написати:

```
#include <include\sys\types.h>
```

Заголовні файли, створені програмістом, зазвичай розташовують у теці проекту. Імена цих файлів у директиві `#include` пишуться у подвійних лапках і завжди з розширенням `h` або `hpp`.

Для створення заголовного файла слід виконати команду Проект/Добавить новый элемент / Заголовочный файл. Буде створено новий модуль, який складатиметься з двох файлів: заголовного файла, наприклад з ім'ям `Header.h`, і відповідного до нього файла реалізації. При зберіганні буде запитано ім'я тільки `cpp`-файла. Ім'я `h`-файла буде створено автоматично.

Заголовний файл може містити:

- оголошення типів `typedef double arr [14];`
- шаблони типів `template <class T> class V { /* ... */ }`
- оголошення (прототипи) функцій `extern int strlen(const char*);`
- визначення функцій-підстановок `inline char get() { return *p++; }`
- оголошення даних `extern int a;`
- визначення констант `const float pi = 3.141593;`
- перерахування `enum bool { false, true };`
- оголошення імен `class Matrix;`
- команди долучення файлів `#include <signal.h>`
- макровизначення `#define Case break;case`
- коментарі `/* Перевірка на кінець файла */`

Перелік того, що саме слід розміщувати в заголовному файлі, не є вимогою мови `C++`, це є лише порада розумного використання долучення файлів.

З іншого боку, в заголовному файлі ніколи не повинно бути:

- визначення функцій `char get() { return *p++; }`
- визначення даних `int a;`
- визначення складених констант `const tb[i] = { /* ... */ };`

Як вже було зазначено, не можна визначити функцію чи то змінну в заголовному файлі, який використовуватиметься кількома файлами програми. Це призведе до помилок повторних визначень. Подібна проблема виникає й тоді, коли помилково залучають один і той самий заголовний файл двічі, наприклад:

```
//Файл app.cpp
```

```
#include "headone.h"
```

```
#include "headone.h"
```

Припустімо, що є файл з кодом програми `app.cpp` і два заголовних файли `head one.h` та `headtwo.h`. До того ж `headone.h` долучає до себе `headtwo.h`. Якщо є потреба залучити обидва заголовні файли до `app.cpp`, слід уважно стежити за тим, щоб не залучити один і той самий файл двічі.

11.2 Завдання - перша частина, задача 1

Задача 1. Створити програму де необхідно реалізувати функцію для виводу двумірного масиву на екран. Сигнатуру функції помістити в заголовочний файл `"printArray.h"`. А вихідний код функції помістити в файл `"printArray.cpp"`. Після чого визвати із головної функції програми (`int main`).

11.3 Хід роботи

11.3.1 Постановка задачі

Дано: `rowLenght`, `columnLenght` - кількість рядків та ліній.

Функціонал:

метод виводу двумірного масиву.

11.3.2 Програмування

Побудова таблиці ідентифікаторів.

Таблиця 11.1 — Таблиця ідентифікаторів

№ з/п	Функціонал	Ідентифікатор
1	<i>кількість рядків</i>	<code>rowLenght</code>
2	<i>кількість ліній</i>	<code>columnLenght</code>
3	<i>n</i>	<code>n</code>
4	<i>m</i>	<code>m</code>
5	<i>метод виводу двумірного масиву</i>	<code>arrayPrint</code>

Створення багатофайлового проекту:

source.cpp – вихідний файл для основної роботи програми;

printArray.hpp – заголовний файл для оголошення функціоналу;

printArray.cpp – вихідний файл для імплементції функціоналу.

Структура програми зображена на рисунку 11.1.

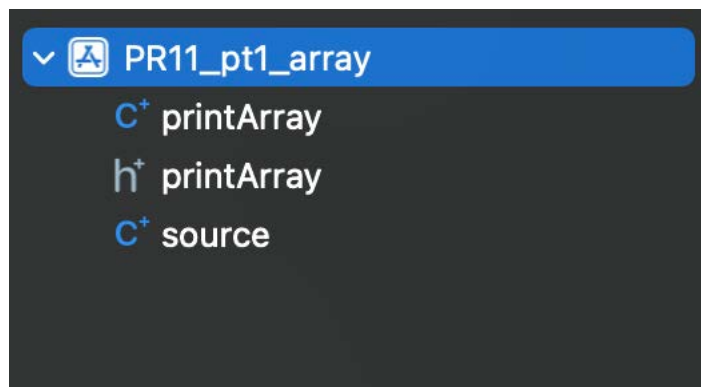


Рисунок 11.1 — Зовнішній вигляд Solution Explorer

Введення тексту програми:

Лістинг 11.1 — Вихідний код файлу printArray.hpp

```
#ifndef PRINTARRAY_HPP
#define PRINTARRAY_HPP
#include <iostream>
using namespace std;
void arrayPrint(int **array, int rowLenght,
int columnLenght);
#endif
```

Лістинг 11.2 — Вихідний код файлу printArray.cpp

```
#include "printArray.hpp"
void arrayPrint(int **array, int rowLenght, int columnLenght) {
    for (int i = 0; i < rowLenght; i++) {
        for (int j = 0; j < columnLenght; j++) {
            cout << array[i][j] << '\t';
        }
        cout << endl;
    }
}
```

Лістинг 11.3 — Вихідний код файлу source.cpp

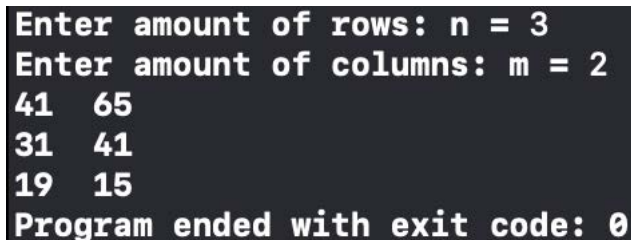
```
#include "printArray.hpp"
#include <iostream>
#include <math.h>
using namespace std;
int main() {
    int rowLenght, columnLenght;
    cout << "Enter amount of rows: n = "; cin >> rowLenght;
    cout << "Enter amount of columns: m = "; cin >>
columnLenght;
    int** A = new int*[rowLenght];
    //заповнення масиву
    for (int i = 0; i < rowLenght; i++) A[i] = new
int[rowLenght];
    for (int i = 0; i < rowLenght; i++) {
        for (int j = 0; j < columnLenght; j++) { A[i][j] =
rand() % 101; }
    }

    arrayPrint(A, rowLenght, columnLenght);

    delete[]A;
    return 0;
}
```

11.3.3 Обчислення, обробка і аналіз результатів

У ході виконання даної роботи отримано наступні результати:



```
Enter amount of rows: n = 3
Enter amount of columns: m = 2
41 65
31 41
19 15
Program ended with exit code: 0
```

Рисунок 11.2 — Результат обчислень

11.4 Завдання - перша частина, задачі 2, 3

Перша частина:

Задача 2. Написати програмний код, який виконує наступні кроки: -створити структуру Point (точка) з полями X та Y (координати) до окремого файлу Point. - розробити функцію для розрахунку відстані між двома точками з іменем getDistance(point, point). - заголовок функції розташувати в заголовочному файлі, код програми в іншому файлі.

та

Задача 3. Створити структуру Circle – коло в середній структурі є два поля: - point (структура Point) - координата на осі координат - radius (типа int або double) - довжина радіусу; - розробити функцію, яка приймає два параметра типу Circle та перевіряє чи перетинаються кола.

11.5 Хід роботи

11.5.1 Постановка задачі

Дано:

- Задача 2: *pointOne.x*, *pointOne.y*, *pointTwo.x*, *pointTwo.y* - координати точок та центри кіл, структура *Point* з координатами точки *x* та *y*;
- Задача 3: *circleOne.radius*, *CircleTwo.radius* - радіуси кіл, структура *Circle* з точкой та радіусом *radius*.

Функціонал: - Задача 2 - метод отримання відстані між точками;
 - Задача 3 - метод перевірки перетину кіл.

11.5.2 Програмування

Побудова таблиці ідентифікаторів.

Таблиця 11.2 — Таблиця ідентифікаторів для Задачі 2

№ з/п	Функціонал	Ідентифікатор
1	<i>перша точка</i>	<i>pointOne</i> (a)
2	<i>друга точка</i>	<i>pointTwo</i> (b)
3	<i>x</i>	<i>x</i>
4	<i>y</i>	<i>y</i>
5	<i>метод отримання відстані</i>	<i>getDistance</i>
6	<i>структура точки</i>	<i>Point</i>

Таблиця 11.3 — Таблиця ідентифікаторів для Задачі 3

№ з/п	Функціонал	Ідентифікатор
1	<i>перша точка</i>	<i>circleOne</i> (a)
2	<i>друга точка</i>	<i>circleTwo</i> (b)
3	<i>point</i>	<i>point</i>
4	<i>radius</i>	<i>radius</i>
5	<i>метод перевірки перетину</i>	<i>circleCross</i>
6	<i>структура кола</i>	<i>Circle</i>

Створення багатофайлового проекту:

source.cpp – вихідний файл для основної роботи програми;

point.hpp, circle.hpp – заголовні файли для оголошення функціоналу;

point.cpp, circle.cpp – вихідні файли для імплементації функціоналу.

Структура програми зображена на рисунку 11.1.

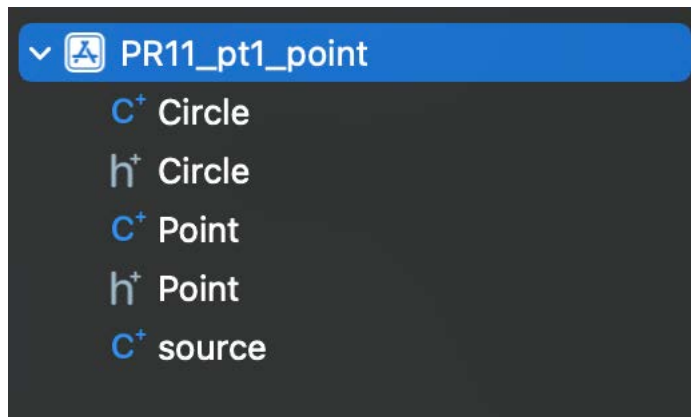


Рисунок 11.3 — Зовнішній вигляд Solution Explorer

Введення тексту програми:

Лістинг 11.4 — Вихідний код файлу Point.hpp Задачі 2

```
#ifndef Point_hpp
#define Point_hpp
#include <math.h>
struct Point {
    int x, y;
};
int getDistance(Point a, Point b);
#endif
```

Лістинг 11.5 — Вихідний код файлу Point.cpp Задачі 2

```
#include "Point.hpp"
int getDistance(Point a, Point b) {
    int oxSquare = (b.x - a.x)*(b.x - a.x);
    int oySquare = (b.y - a.y)*(b.y - a.y);
    return sqrt(oxSquare+oySquare);
}
```

Лістинг 11.6 — Вихідний код файлу Circle.hpp Задачі 3

```
#ifndef Circle_hpp
#define Circle_hpp
#include "Point.hpp"
struct Circle {
    Point point;
    int radius;
};
bool circleCross(Circle a, Circle b);
#endif
```

Лістинг 11.7 — Вихідний код файлу Circle.hpp Задачі 3

```
#include "Circle.hpp"
bool circleCross(Circle a, Circle b) {
    if ((a.radius + b.radius) < getDistance(a.point, b.point))
        return false;
    else if (a.radius + getDistance(a.point, b.point) < b.radius ||
b.radius + getDistance(a.point, b.point) < a.radius) return false;
    return true;
}
```

Лістинг 11.8 — Вихідний код файлу source.cpp Задач 2, 3

```
#include "Point.hpp"
#include "Circle.hpp"
#include <iostream>
using namespace std;
int main() {
    Point pointOne;
    Point pointTwo;
    cout << "Enter coordinates of first point." << endl;
    cout << "x: "; cin >> pointOne.x;
    cout << "y: "; cin >> pointOne.y;
    cout << "Enter coordinates of second point." << endl;
    cout << "x: "; cin >> pointTwo.x;
    cout << "y: "; cin >> pointTwo.y;
    cout << "Distance between points is " << getDistance(pointOne,
pointTwo) << endl;

    Circle circleOne;
    Circle circleTwo;
    circleOne.point = pointOne;
    circleTwo.point = pointTwo;
    cout << "Radius of first circle: "; cin >> circleOne.radius;
    cout << "Radius of second circle: "; cin >> circleTwo.radius;

    if (circleCross(circleOne, circleTwo) == false) {
        cout << "Circles aren't cross." << endl;
    } else cout << "Circles are cross." << endl;

    return 0;
}
```


11.5.3 Обчислення, обробка і аналіз результатів

У ході виконання даної роботи отримано наступні результати:

```
Enter coordinates of first point.  
x: 2  
y: 2  
Enter coordinates of second point.  
x: 3  
y: 4  
Distance between points is 2  
Radius of first circle: 1  
Radius of second circle: 5  
Circles aren't cross.  
Program ended with exit code: 0
```

Рисунок 11.4 — Результат обчислень

11.6 Завдання - друга частина

Створити заголовний файл з функціоналом лабораторної роботи № 6 та 8.

Лістинг 11.9 — Вихідний код файлу room.hpp лабораторної роботи № 6.1

```
#ifndef volume_hpp  
#define volume_hpp  
#include <iostream>  
using namespace std;  
struct Room {  
    float x;  
    float y;  
    float z;  
};  
float calc(Room a); //обчислення (x*y*z)  
void comparison (Room a, Room b, Room c); //порівняння приміщень  
#endif
```

Лістинг 11.10 — Вихідний код файлу note.hpp лабораторної роботи № 6.2

```
#ifndef note_hpp  
#define note_hpp  
#include <string>  
struct NOTE {  
    std::string name;  
    long tel;  
    int bday[3];  
};  
NOTE setPersonData(int n, NOTE blocknote[n]);  
bool bdayChecker (int i, int month, NOTE blocknote[i]);  
#endif
```

Лістинг 11.10 — Вихідний код файлу Array.hpp лабораторної роботи № 8.1

```
#ifndef Array_hpp
#define Array_hpp
#include <iostream>
using namespace std;
void inputArray(int *Arr, int n); //automatic mode
void showArray(int *Arr, int n); //вивод масиву
void inputArray(int*Arr, int n, bool mode); //manual mode
bool orthogonalityCheck(int * arrx, int * array, int n); //arrx * array = 0
#endif
```

Лістинг 11.11 — Вихідний код файлу Array.hpp лабораторної роботи № 8.2

```
#ifndef Matrix_hpp
#define Matrix_hpp
#include <stdio.h>
#include <iostream>
using namespace std;
void inputMatrix(int**matrix, int n); //manual mode
void inputMatrix(int**matrix, int n, bool mode); //automatic mode
void showMatrix(int **matrix, int n); //вивод двумірного масиву
int getLargestElement(int **matrix, int n); //найбільший елемент
void devideByLargestElement(int **matrixA, int **matrixB, int n, int
max); //B[i][j] = A[i][j] / max;
#endif
```

11.5 Програми та обладнання.

Xcode, Pages

11.6 Висновки.

Під час виконання цієї практичної роботи були здобуті практичні навички роботи зі створення та використання заголовного файлу.